Assignment 2: Quantum Teleportation

Lingyu Gong

I. Lab purpose

Get hands-on experience with quantum algorithms by conducting specific experiments and following the entire process to build and test the algorithm. This experiment will show how quantum information transfer between users, go deeper to what is entanglement and how it works.

II. Teleportation Algorithms

1. Definition

Quantum teleportation is a technique used to transfer the entire state of one quantum system to another, even if they are separated by a large distance, similar to a protocol. It is a process that enables the transmission of quantum information from one location to another without the need for physical movement.

2. Background Introduction

We assume there are two characters named Alice and Bob, who are not real human beings, just like two entities. Now Alice (as a sender) wants to send a message to Bob (as a receiver), and the format of the message is qubit. So, the main issue is how to transfer quantum information.

3. Set-up

Originally, Alice and Bob had their own qubits A and B, which can be put together as state |+> and this can present an entanglement state. Now, Alice wants to send qubit Q to Bob, however, in the real world it's still a challenge to send qubit to others and if we use a classical information channel to achieve this goal, it will be easier to implement, but we are not going to use classical channel alone because it's easy to cause information leakage, that's the reason to set two initial qubits at first.

III. Lab Implement

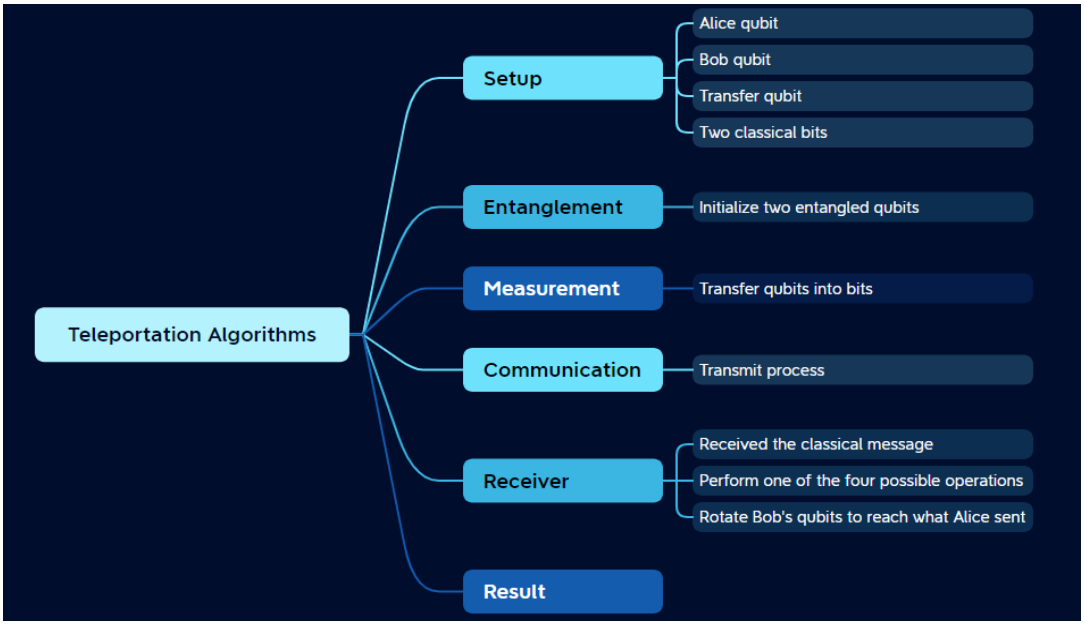

Figure 1 The Structure of implement the algorithm.

To be more specific, the whole program is as below.

1. Initialize all qubits and bits that we need.

```
# Initialize register for 3 qubits, Alice and Bob have a initial one, another for transfer.
qr = QuantumRegister(3, name="q")
# Initialize register for classical bits
crz, crx = ClassicalRegister(1, name="crz"), ClassicalRegister(1, name="crx")
# One register for Bob's receives
crb = ClassicalRegister(1, name="crb")
# Set up teleportation circuit
teleportation_circuit = QuantumCircuit(qr, crz, crx, crb)
```

2. We need to entangle A and B.

```
def create_bell_pair(qc, a, b):
    # Put qubits into state |+>
    qc.h(a)
    # cnot: a as control; b as target
    qc.cx(a, b)
```

3. Time for Alice. Define the qubit psi (want to send) with the help by a (Alice hold at first)

```
def alice_gates(qc, psi, a):
    # The qubit that going to be send
    qc.cx(psi, a)
    # Use Hadamard gate: 50% 0 50% 1
    # Create entangled states
    qc.h(psi)
```

4. Transmit time. Change into bits and send.

```
# measure two qubits from Alice
# Store them into classical bits
def measure_and_send(qc, a, b):
    qc.barrier()
    # Restore them into classical bits
    qc.measure(a, 0)
    qc.measure(b, 1)
```

5. Time for Bob. Receive classical bits to perform one of four possible operations on his entangled qubit.

```
def bob_gates(qc, qubit, crz, crx):
    # conditional logic is used to control the application of gates.
    # allows to apply gates depending on whether the classical bit is set to 1 or 0.
    # qc.if_test to control gates by classical bits
    with qc.if_test((crx,1)):
        # The X gate is a quantum gate that performs a bit-flip operation
        qc.x(qubit)
    with qc.if_test((crz, 1)):
        # The Z gate is a quantum gate that performs a phase-flip operation.
        qc.z(qubit)
```

6. Bob's measure. According to the message sent by Alice, Bob needs to figure out what is the outcome that Alice want to send.

```
crb = ClassicalRegister(1, name="crb")
teleportation_circuit.measure(2, 2)
```

IV. Outcome explanation

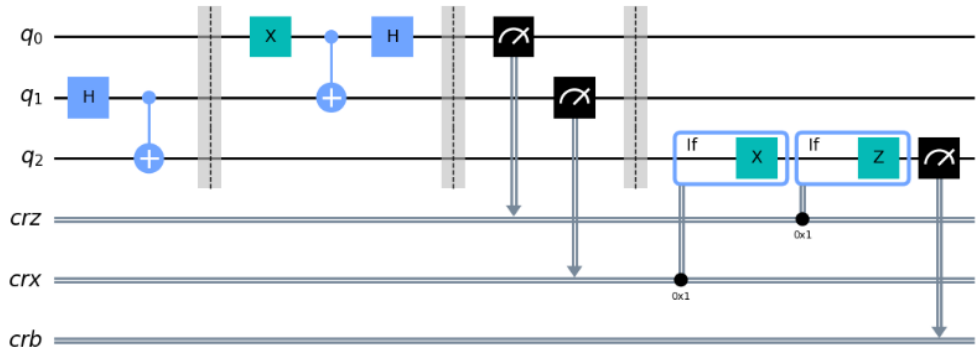

Figure 2 The whole circuit diagram

The circuit above is the vitalization version as I described in the previous chapters and the ideal result should be like this:

If q0 = 1, crb = 1 ;
If q0 = 0, crb = 0 ;

Because "q0" is the message Alice wants to send to Bob and "crb" is the value that shows what Bob received. They should be the same.

The result to see whether it's correct or not will be shown below. I use the resource from my IBM account for instance: "ibm-q/open/main" and name "ibmq_qasm_simulator" to show me the data. The code is as below.

```
from qiskit_ibm_provider import least_busy
from qiskit_ibm_provider import IBMProvider
from qiskit.tools.monitor import job_monitor
from qiskit import *
# load backend
provider = IBMProvider()
backend = provider.get_backend(backend_name, instance=hgp)

#backend = least_busy(provider.backends(dynamic_circuits=True))
print(f"Using backend {backend.name}")

# transpile our circuit for the selected backend
backend.target.add_instruction(IfElseOp, name="if_else")
transpiled_circuit = transpile(teleportation_circuit, backend)

# run job
shots = 1000
job = backend.run(transpiled_circuit, shots=shots, dynamic=True)

print(f"Job ID: {job.job_id()}")
job_monitor(job)  # displays job status under cell
# Get the results and display them
exp_result = job.result()
exp_counts = exp_result.get_counts()
plot_histogram(exp_counts)
```

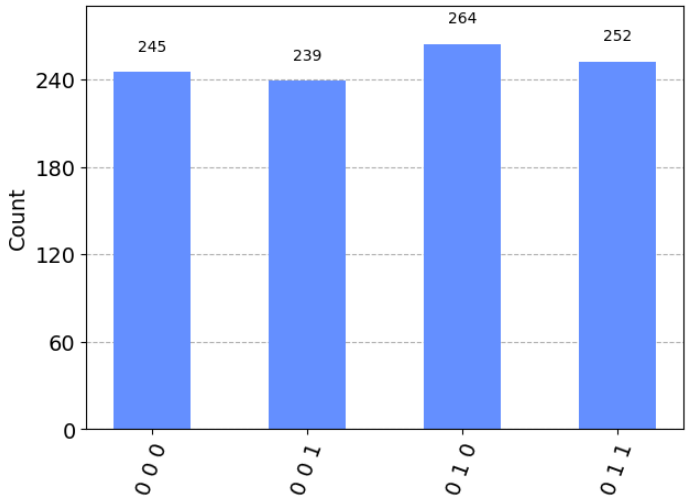

Figure 3 The result diagram as "q0=0"

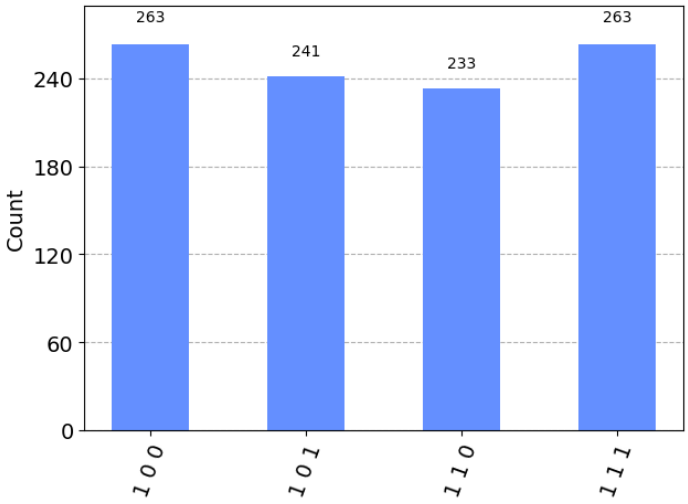

Figure 4 The result diagram as "q0=1"

As the result shows, when q0 = 0, crz = 0, and q0 = 1, crz = 1.

V. Summary

Throughout the course of this project, I gained knowledge on the transfer of quantum information. Specifically, I learned about the significance of entanglement and the transformation of quantum information into classical information for transfer. Additionally, I have summarized the fundamental steps involved in building and testing a quantum algorithm, which include understanding the problem, selecting an appropriate quantum algorithm, designing a quantum circuit, implementing error correction and noise mitigation, quantum compilation, testing on a quantum simulator, executing on quantum hardware, iterative improvement, and documentation and reporting. Although not all these steps were utilized in this lab, I acquired a lot of valuable insights during the entire process.